Next Generation Integration Scorecard
Tracking Functional Area

The Tracking Functional Area provides a means for tracking Issues. The Tracking Functional Area is part of a functional cluster that includes the Resourcing Functional Area, Ontology Functional Area, Workflow Functional Area, and Process Functional Area. Each of these areas defines a set of operations that define the overall work management system.

Issue

Issues are relationships between a customer and a Queue. The effective dates of the Issue represent the opening and closing of the Issue. An Issue defines some built-in states indicating if the Issue is resolved, waiting for a response, or blocked on another Issue. Additional states can be modeled within a system orchestrating the Process Functional Area or through the Workflow Functional Area. Issues can be related to each other in various ways.
Also known as: Ticket, Work, Request, Jira

❏ Read Operations for *Issues:*   *Score*

❏ Get Issues given unique Ids	
❏ Get all the Issues in a system or Catalog	
❏ Query (Search) Issues based on attribute-based query terms or keywords	
❏ Register for notifications Issues have been created, updated or deleted	

❏ Write Operations for *Issues:*

❏ Create, Update and Delete Issues	
❏ Add an alias Id to reference an existing Issue	
❏ Close and Reopen Issues	
❏ Organize Issues into Catalogs	

❑ Operations for retrieving one's own *Issues:*

❑ Get open Issues where the authenticated Agent is the customer	
❑ Get the list of open issues in progress for the Agent	
❑ Get closed Issues where the authenticated Agent is the customer	
❑ Get the list of issues opened since given date for the Agent	
❑ Get all issues for the Agent	

❑ Operations for tracking *Issues* in a Queue*:*

❑ Get an Issue specified by its Id	
❑ Get the list of all open issues in a given Queue	
❑ Get the list of all open unassigned issues in a given Queue	
❑ Get the list of open issues in a given Queue that has been idle since the given date.	
❑ Get the list of open Issues in a given Queue assigned to the given resource	
❑ Get the list of Issues by [[topic]] for a given Queue which may include issues of any child [[topic]]	
❑ Get the list of closed Issues that were created during a given date range for a given Queue	
❑ Get the list of all Issues that were in an open status on a given date for a given Queue	
❑ Get the list of all Issues created since the given date for a given Queue	

❑ Operations for resourcing *Issues:*

❑ Get the list of resource assignable for a given Queue	
❑ Get the list of candidate resources from the Queue available for a given Issue sorted from best suited	
❑ Get the list of Issues assigned to a given Resource	
❑ Get the list of Issues assigned to a given Resource in a given Queue	
❑ Assign an Issue to a Resource. The resource must be a valid resource in the issue queue	
❑ Unassign an Issue from a Resource	

❑ Operations for triaging and routing *Issues:*

❑ Move a given Issue to another given Queue	
❑ Get the list of assignable topics for a given Queue.	

❏ Assign a given topic (ontology Subject) to a given Issue	
❏ Close a given Issue	

❏ Operations for inspecting *Issue* relationships*:*

❏ Get any subtasks for a given Issue	
❏ Get any duplicate linked Issues for a given Issue	
❏ Get any branched issues off of a given Issue	
❏ Get any blocking Issues of a given Issue	
❏ Get any blocked Issues as a result of a given Issue	

❏ Operations for managing *Issue* relationships*:*

❏ Create a nested subtask Issue	
❏ Add and remove duplicate links between Issues	
❏ Branch an Issue	
❏ Copy a LogEntry from the root Issue to be visible in a given branched Issue	
❏ Remove a LogEntry from a given branch Issue.	
❏ Add or remove blocking relationships between Issues	

Minimally Supported Attributes of *Issues:*

● A unique and permanent identifier
● The name of the Issue
● A description of the Issue
● The type of the Issue: task, bug, feature, request, complaint
● The Queue associated with this Issue
● The customer associated with this Issue
● The topic of this Issue, managed vocabulary as can be expressed as an ontology Subject
● The master issue, If this Issue is a subtask of another issue
● Duplicate issues, if this Issue is a duplicate of other issues
● The branched issue, if this Issue is a branch of another issue
● The root issue, if this Issue is a branch of another issue
● The priority Type of this Issue
● The creator Resource of this issue

● The authenticated Agent that created this Issue
● Boolean test if this Issue has been reopened after a close
● The reopener Resource of this issue
● The authenticated Agent that reopened this Issue
● The last reopened date
● The due date, if applicable
● Boolean test if this issue is pending a response from the customer
● Boolean test if this issue is blocked on other Issues
● The blocking Issues, if this Issue is blocked on other Issues
● Boolean test if this Issue is resolved. If the Issue was reopened, it is no longer resolved
● The resolver Resource of this Issue
● The authenticated Agent that resolved this Issue
● The resolved date. A resolved Issue is still open until it is closed
● A type indicating the resolution; "fixed," "canceled", "cannot reproduce", etc
● Boolean test if this issue is closed. An issue may be left opened after being resolved for acknowledgement or review
● The Resource of the closer
● The authenticated Agent that closed the Issue
● The closed date
● Boolean test if this issue is assigned
● The assigned Resource
● The effective start date and end date of this Issue
● The reason this Issue was closed, if applicable

LogEntry

Issues can have logs to provide a record of actions and comments.
Also known as:

❏ Read Operations for *LogEntries:*   *Score*

❏ Get LogEntries given unique Ids	

❏ Get all the LogEntries in a system or Catalog	
❏ Query (Search) LogEntries based on attribute-based query terms or keywords	
❏ Register for notifications that LogEntries have been created or updated	

❏ Operations for *LogEntry* Issue Messaging*:*

❏ Create, Update and Delete comments, as LogEntries, on an Issue	
❏ Delete comments for catalog, data range, Queues and Issues	
❏ Send a customer a message and records it in the issue log	

Minimally Supported Attributes of *LogEntries:*

● A unique and permanent identifier
● The name of the LogEntry
● A description of the LogEntry
● The type of the LogEntry: state change, assignment change, comment
● The Agent associated with this LogEntry
● The Issue associated with this LogEntry
● The date of this LogEntry
● The Action of this LogEntry as defined in a controlled vocabulary
● The summary text header for this LogEntry
● The text message of this LogEntry

Queue

A Queue governs the constraints and processing of a set of Issues. All Issues belong to one Queue.
Also known as:

❏ Read Operations for *Queues:*   *Score*

❏ Get Queues given unique Ids	
❏ Get all the Queues in a system or Catalog	
❏ Query (Search) Queues based on attribute-based query terms or keywords	
❏ Register for notifications that Queues have been created or updated	

❏ Write Operations for *Queues:*

❏ Create, Update and Delete Queues	
❏ Add an alias Id to reference an existing Queue	
❏ Organize Queues into Catalogs	

❏ Operations for resourcing *Queues:*

❏ Get the list of resources assigned in a given Queue	
❏ Assign a Resource to a given Queue	
❏ Unassign a Resource from a given Queue	

Minimally Supported Attributes of *Queues:*

● A unique and permanent identifier
● The name of the Queue
● A description of the Queue
● The type of the Queue
● The provider of this Queue
● Any available branding for this Queue, for example an organizational logo or thumbnail
● Boolean indicator capable of suspending and resuming the Queue

Catalog

A directory or other kind of organization for learning objectives. Such a grouping serves to separate Queues managed by different organizations.
Also known as: FrontOffice

❏ Read Operations for *Catalogs:*   *Score*

❏ Get Catalogs given unique Ids	
❏ Get all the Catalogs in a system or Catalog	
❏ Query (Search) Catalogs based on attribute-based query terms or keywords	
❏ Traverse Hierarchical relationships between Catalogs	
❏ Register for notifications Catalogs have been created, updated or deleted	

❏ Register for notifications that hierarchical relationships between Catalogs have been created, updated or deleted	

❏ Write Operations for *Catalogs:*

❏ Create, Update and Delete Catalogs	
❏ Add an alias Id to reference an existing Catalog	
❏ Manage Hierarchical relationships between Catalogs	

Minimally Supported Attributes of *Catalogs:*

● A unique and permanent identifier.
● The name of the Catalog
● A description of the Catalog
● The type of the Catalog
● The provider of this Catalog
● Any available branding for this Catalog, for example, an organizational logo
● Any licensing (terms of usage) associated with this Catalog